テンソルデータモデルを用いた多次元データマイニング

横林　亮平†　　三浦　孝夫†

† 法政大学大学院 理工学研究科システム理工学専攻　〒 184–8584 東京都小金井市梶野町 3-7-2
E-mail: †ryohei.yokobayashi.2j@stu.hosei.ac.jp, ††miurat@hosei.ac.jp

あらまし　本研究では，テンソルデータモデルの枠組みの中で同時関係抽出を行うデータ操作手法を提案する．テンソルデータモデルを用いることで従来の複雑なアルゴリズムを簡素化し，単純なデータ操作となる．一般的に同時関係抽出は，同じレコード (トランザクション) 内の強い相関集合を抽出するが，ここでは別トランザクションを跨いだ関係の獲得を実現する．このとき既存手法では，データ記述長が大きくなることが問題となるがテンソルデータモデルを用いることで記述長の削減が可能となることを示す．

キーワード　Tensor，Data Model，Association Rules，Data Minig

1. 前　書　き

近年，センサデバイスや半導体メモリの技術発展に伴い取得可能なデータ量とその種類が増加している．これら膨大なデータから有用なデータを効率的に抽出するデータマイニング技術がますます重要となっている．データマイニング研究の中でも同時関係抽出 [1] は代表的な技法の 1 つである．現在は，トランザクションデータベースから複数のトランザクションにおいて頻出するアイテム集合を見つける手法が一般的となっている．しかし，より有益な情報はトランザクションに跨って頻出する同時関係 [8] であると言える．例えば一般的な同時関係は，

A 社の株価が上がれば ⇒ 同時に B 社の株価が上がる

といった関係ルールである．しかし，本稿では

A 社の株価が上がれば ⇒ 2 日後に B 社の株価が上がる

という時間などの距離概念を追加し，トランザクションを越えたルール (多次元同時関係) 抽出 [8] を行う．ここでは，1 つの距離概念に止まらず複数の意味の保持を行うため，複数の軸 (階) を有するデータ構造を考える．

本研究ではテンソルデータモデル [12] を導入する．テンソルデータモデルを用いる利点は以下にある．複数の階の意味を同時に考えることができるため，トランザクションを跨ぐルール抽出ではより具体的な意味を有するアイテム集合を抽出できる．これによりデータの移り変わりなどの予測がより具体的になる．また，テンソルデータモデルでは階や次元を柔軟に変更できる．例えば，日単位のデータから月単位のデータに考え直す際，次元の統合などによるデータの変換操作が行える．既存の複雑なアルゴリズムを用いた手法も単純なテンソルデータ操作で抽象化することができる．本稿は，テンソルデータの論理演算などデータマイニング操作を定義し，これらを用いて既存の複雑なアルゴリズムの抽象化を行う．

本研究の貢献点は以下の 3 点である．

- テンソルデータモデルのデータマイニング操作の提案
- テンソルモデルの枠組みで多次元同時関係抽出の実現
- 多次元同時関係抽出を行う際のメモリ消費の考慮

2 章以降では，以下のように構成される．2 章ではテンソルデータモデルのデータ記述とその操作について述べ，3 章では同時関係抽出手法について述べる．特に同時関係抽出の代表手法である Apriori [1] とその関連研究 [8] について言及する．4 章ではテンソルデータモデルによる同時関係抽出操作について述べる．5 章では，実験により提案手法の有用性を示し，6 章で結論とする．

2. テンソルデータモデル

2.1 高階データ表現

高階（テンソル）データ [12] とは，階数が 3 以上となるデータ構造を指す．テンソルは $X \in \mathcal{T}[I_1 \times I_2 \times ... \times I_N]$ で表し，N をこのテンソルの階数という．また，ベクトルと行列はそれぞれ 1 階のテンソル，2 階のテンソルと呼びテンソルで一般化することで同様に扱える．高階テンソルを用いることで，行列データよりも詳細な情報表現を行うことができ，またデータ操作が単純になる．例えば，ある店の商品売上情報をみる場合，行列では 1 年ごとといったまとまったデータ情報を表現することになる．しかし，テンソル表現をする場合，年間売上を月別に表示することができる．図 1 にテンソルのデータ例を示す．

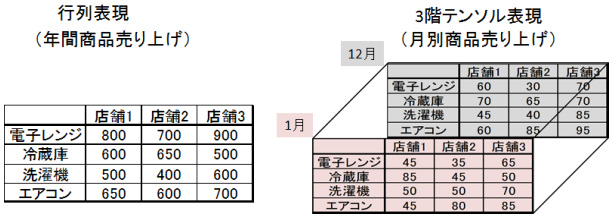

図 1　店舗 1,2 の商品売り上げ

2.2 テンソルデータ操作

テンソルデータ操作 [6] [12] には，テンソル射影，テンソル選択，テンソルどうしの和・差・積演算，階変換，次元縮小，次元復元，検索操作などがある．また本稿では，データ要素別ベクトル化，テンソルのデータ統合などを行う論理演算，シフト操作，要素のカウント操作を導入する．以下，個々の操作を定義する．

射影は π と記述し，指定した条件に従い部分的なテンソルを取り出す．例えば，図 1 のテンソルの月別売上データにおいて，電子レンジと洗濯機が 12 月以外に売れた店舗情報を取り出す．この操作は，$X' = \pi_{(\sharp 1(\text{電子レンジ, 洗濯機})\sharp 2(*)\sharp 3(12\ 月)^{-1})}(X)$ で表す．

選択は σ で記述し，限定的にデータの一部分を抽出する．例えば，図 1 のテンソルデータより洗濯機が 70 台以上売れた店を抽出する．データ操作は，$X'' = \sigma_{\sharp 2.\text{洗濯機}>70}(X)$ となる．射影，選択によって得られるデータ例を図 2 に示す．

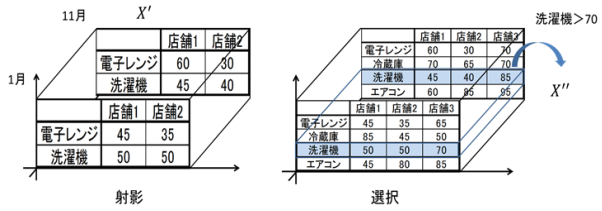

図 2　射影と選択操作

次に，テンソルの演算操作について述べる．階数と次元数の等しい 2 つのテンソル $X, Y \in \mathcal{T}[I_1 \times I_2 \times \ldots \times I_N]$ についてそれぞれの要素を $x_{i_1,i_2,\ldots,i_N}$，$y_{i_1,i_2,\ldots,i_N}$ と表す．2 テンソルの和と差は，$X+Y$，$X-Y$ と表し，その要素は

$$X + Y = x_{i_1,i_2,\ldots,i_N} + y_{i_1,i_2,\ldots,i_N}$$
$$X - Y = x_{i_1,i_2,\ldots,i_N} - y_{i_1,i_2,\ldots,i_N}$$

となる．2 つのテンソルの積は

$$\langle X, Y \rangle = \sum_{i=1}^{I_1} \sum_{i=2}^{I_2} \cdots \sum_{i=N}^{I_N} x_{i_1,i_2,\ldots,i_N} y_{i_1,i_2,\ldots,i_N}$$

となる．

テンソルは，行列やベクトルの集合であり，ベクトル表現ができる．ベクトル表現変換を形式化したファイバとモードについて言及する [6]．ファイバは，テンソルをベクトル表現したものである．3 階のテンソルを例に挙げる．このとき，ベクトル表現方法は 3 種類存在する．3 種の表現を図 3 に示す．固定したテンソルに対して，列方向 (column)，行方向 (row)，奥行き方向 (tube) にファイバをとり，それぞれモード 1 ファイバ，モード 2 ファイバ，モード 3 ファイバと呼ぶ．テンソルを行列

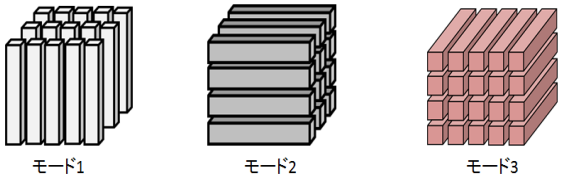

図 3　ファイバとモード

で表現するときは，n-モード行列化と呼びモード n のファイバを用いて行列化する．n-モード行列化を図 4 に示す．モード n

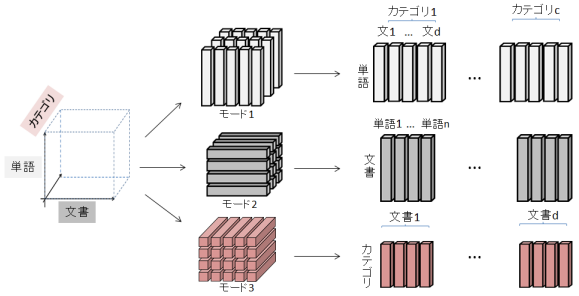

図 4　n-モード行列化

のファイバから構成された n-モード行列は，$X_{(n)}$ で表現する．

テンソルに対する情報検索は，モード n ファイバまたは n-モード行列化を用いて，検索質問との類似度を求めることにより行う．検索質問は，索引語の集合からなり，類似度は一般的に内積や余弦類似度を用いる．したがって，テンソルとベクトルの内積 (ベクトルテンソル積) は類似度操作の意味を持つ．テンソル $X \in R^{I_1 \times I_2 \times \ldots \times I_N}$ とベクトル $V \in R^{I_n}$ とすると，ベクトルテンソル積は $Y_{(n)} = X \times_n V \in \mathcal{T}[I_1 \times I_2 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N]$ であり，

$$X \times_n V = ((Y_{i_1,..,i_{n-1},i_{n+1},..,i_N})) \quad (1)$$

$$Y_{i_1,..,i_{n-1},i_{n+1},..,i_N} = \sum_{i_n=1}^{I_n} X[i_1,..,i_{n-1},i_n,i_{n+1},..,i_N] V[i_n] \quad (2)$$

となる．このとき，Y は X に対して階数が 1 つ低くなる．

テンソル (図 6) に対してベクトルテンソル積を行う．ベクトル $(1,\ 2,\ 3)^t$ のとき，ベクトルテンソル積による類似度操作を以下に示す．

$$\begin{bmatrix} 45 & 35 & 65 & 60 & 30 & 70 \\ 85 & 45 & 50 & 70 & 65 & 70 \\ 50 & 50 & 70 & 45 & 40 & 85 \\ 45 & 80 & 85 & 60 & 85 & 95 \end{bmatrix} \times \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 310 & 210 \\ 325 & 410 \\ 360 & 380 \\ 1160 & 535 \end{bmatrix}$$

要素が内積となるため，要素の高い部分の属性データをランキングし表示することがテンソル情報検索となる．テンソルにおける情報検索操作の例を図 5 に示す．

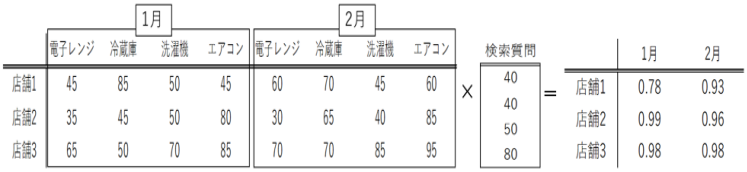

図 5　テンソル情報検索

次元縮小はデータを低次元に射影し，データの空間量と複雑さを大幅に削減することができる．テンソルデータモデルにおいて，行列とテンソルの内積 (行列テンソル積) 操作は次元縮小を意味する．テンソル $X \in \mathcal{T}[I_1 \times I_2 \times \ldots \times I_N]$

，行列 $M \in R^{J_n \times I_n}$ とし，行列テンソル積を Y とする．$Y_{(n)} = X \times_n M \in \mathcal{T}[I_1 \times I_2 \times ... \times I_{n-1} \times J_n \times I_{n+1} \times ... \times I_N]$，$j_n = 1, .., J_n$ とおくと，

$$X \times_n M = ((Y_{i_1,..,i_{n-1},j_n,i_{n+1},..,i_N})) \tag{3}$$

$$Y_{i_1,..,i_{n-1},j_n,i_{n+1},..,i_N} = \sum_{i_n=1}^{I_n} X[i_1,..,i_{n-1},i_n,i_{n+1},..,i_N]M[j_n,i_n] \tag{4}$$

となる．$J_n \leqq I_n$ であれば次元数は小さくなる．$X \in \mathcal{T}[2\times3\times2]$ のテンソル (図 6) を次元縮小する場合について述べる．行列

	1月			2月		
	店舗1	店舗2	店舗3	店舗1	店舗2	店舗3
電子レンジ	45	35	65	60	30	70
冷蔵庫	85	45	50	70	65	70
洗濯機	50	50	70	45	40	85
エアコン	45	80	85	60	85	95

図 6　1,2 月店舗売り上げ

$M \in R^{3\times4}$ を用いると，

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 3 & 2 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 45 & 35 & 65 & 60 & 30 & 70 \\ 85 & 45 & 50 & 70 & 65 & 70 \\ 50 & 50 & 70 & 45 & 40 & 85 \\ 45 & 80 & 85 & 60 & 85 & 95 \end{bmatrix} = \begin{bmatrix} 320 & 385 & 445 & 340 & 400 & 525 \\ 335 & 245 & 365 & 365 & 260 & 435 \end{bmatrix}$$

となり，次元数が小さくなる．

$Y_{(n)} = X \times_n M$ は以下の特性を持つ．

$$X \times_m A \times_n B = X \times_n B \times_m A (m \neq n)$$
$$X \times_n A \times_n B = X \times_n (BA)$$

また，特異値分解と Random Projection をテンソルに拡張した HOSVD [4] と高階ランダムプロジェクション (HORP) [11] [12] による次元縮小がある．これらの手法は次元縮小したときの誤差保証がなされておりデータ構造を保ったまま次元縮小が可能となる．HOSVD と HORP の様子を図 7 に示す．

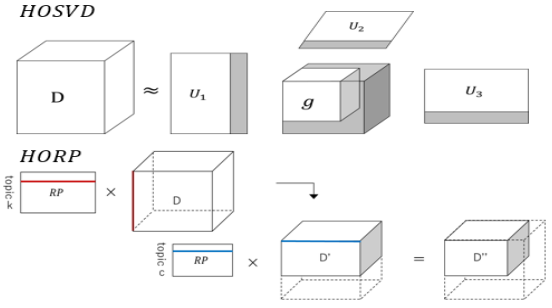

図 7　HOSVD と HORP による次元縮小

2.3　データマイニング操作

データマイニング操作として要素別ベクトル化，論理演算，シフト演算，要素カウント操作を提案する．

テンソルのセル内には，複数の要素を保持する可能性があるためベクトルで一般化される．要素別ベクトル化操作は，軸基点のデータから各要素へと見方を変え，他のセルにおける要素との関係を捉え易くする．テンソル $X \in \mathcal{T}[I_1 \times I_2 \times ... \times I_N]$，各要素 $x_{i_1} = x_{i_{1,1},i_{1,2},...,i_{1,N}}$ において要素ベクトル化を行うと，$\vec{x}_{i_{N,1}} = x_{i_{1,1},i_{2,1},...,i_{N,1}}$ となる．図 8 に要素別ベクトル化を示す．

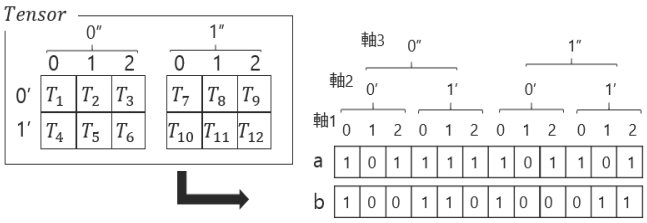

図 8　要素別ベクトル化

テンソルの論理演算について言及する．テンソルの論理演算では，次元の統合や共通項の抽出を行う．2 つのテンソル X, Y において要素が 2 進数の場合，共通項抽出 (AND) 操作 $z_{i_{1,1}} = (x_{i_1}\ AND\ y_{i_1})$ は，
$(x_{i_1} = 1 \wedge y_{i_1} = 1)$ のとき $z_{i_{1,1}} = 1$
$(x_{i_1} \neq 1 \vee y_{i_1} \neq 1)$ のとき $z_{i_{1,1}} = 0$
で表す．また次元の統合操作 (OR) は，
$(x_{i_1} = 1 \vee y_{i_1} = 1)$ のとき $z_{i_{1,1}} = 1$
$(x_{i_1} = 0 \wedge y_{i_1} = 0)$ のとき $z_{i_{1,1}} = 0$
となる．AND，OR 操作は別のテンソルどうしだけでなく同じテンソル内の別次元でも同様の操作が可能である．
$z_{i_{1,2}} = (x_{i_1}\ AND\ x_{i_2})$ は，
$(x_{i_1} = 1 \wedge x_{i_2} = 1)$ のとき $z_{i_{1,2}} = 1$
$(x_{i_1} \neq 1 \vee x_{i_2} \neq 1)$ のとき $z_{i_{1,2}} = 0$

要素が 2 進数以外の場合，$AND_{(min)}, AND_{(max)}, AND_{(average)}$, $OR_{(min)}, OR_{(max)}, OR_{(average)}$ が存在する．$AND_{(min)}$ では，テンソル内の要素 n,m に対して $n \geqq m$ が成り立つとすると，
$(x_{i_1} = n \wedge y_{i_1} = m)$ のとき $z_{i_{1,1}} = m$
を得る．$AND_{(max)}$ では $z_{i_{1,1}} = n$ を $AND_{(ave)}$ では $z_{i_{1,1}} = \frac{n+m}{2}$ を得る．$(x_{i_1} = n, y_{i_1} = m)$ において $OR_{(min)}, OR_{(max)}, OR_{(average)}$ でそれぞれ
$z_{i_{1,1}} = m$, $z_{i_{1,1}} = n$, $z_{i_{1,1}} = \frac{n+y}{2}$ となり，$m = 0$ のとき
$z_{i_{1,1}} = n$, $z_{i_{1,1}} = n$, $z_{i_{1,1}} = \frac{n}{2}$ を得る．

テンソルシフト演算操作は，条件に従い要素をシフトする操作である．$x_{i_1,i_2,..,i_N}$ に対して右シフトを行うと $x_{0,i_1,i_2,..,i_{N-1}}$ が得られる．テンソルをデータ要素ベクトル化や行列化を用いることで各ベクトル・行列に対してもシフト演算が可能となる．データ要素ベクトル化と右シフト演算操作の様子を図 9 に示す．次に要素が全て 1 のテンソル Y との内積について言及する．例として 2 階のテンソル X と 1 階のテンソル Y の内積を考える．$X^{n\times m} \times Y^{m\times1} = (Z^{n\times1})$ であるため，

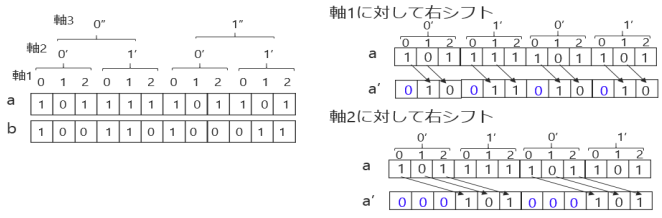

図 9　テンソルシフト操作

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^t = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

を得る．全要素 1 のベクトルとの内積計算を行う場合，各次元のカウント操作を行うことと等価となる．

3. 同時関係抽出

同時関係を積極的に利用するアプリケーションに，スーパーマーケットの売り上げデータを分析するマーケットバスケット分析などがある．これらのデータベースはトランザクションデータベースと呼ばれ，トランザクションレコードが含まれる．同時関係は，同じトランザクション中で同時に起こる (出現する) 集合の内，その集合が別の複数トランザクションでも含まれる高出現頻度集合をルールとして抽出する．Agrawal [1] らにより提案された Apriori では，このとき候補となる集合の絞り込みを行うことで効率的にルールを取得する．

3.1 Apriori

アイテム集合を $I = \{i_1, i_2, \cdots, i_s\}$，トランザクションデータベース D をトランザクション $\{t_1, t_2, \cdots t_n\}$ の集合とする．トランザクション $t_i\,(i = 1, 2, \cdots, n)$ は I のサブセットからなる．トランザクションとアイテム集合の様子を表 1 の左に示す．トランザクション内に同時に出現するアイテム項目集合の内，高出現頻度のものをルールとして取得する．例えば a,b が共起するとき，

$a \Rightarrow b$

といった同時関係ルールを得る．出現頻度の多さは支持度 (support) と確信度 (confident) をユーザ定義の閾値と比較することにより計る．

$$support = \frac{\text{条件 } X, Y \text{ を同時に満たすトランザクション}}{\text{全トランザクション}}$$

$$confident = \frac{\text{条件 } X, Y \text{ を同時に満たすトランザクション}}{\text{条件 } X \text{ を満たすトランザクション}}$$

ユーザ定義の閾値は最小支持度 (min-support)，最小確信度 (min-confidence) としこれらを上回るものを頻出項目集合とし抽出する．Apriori は全てのアイテムの組み合わせを考えるのではなく，長さ $(k-1)$ の頻出項目集合から長さ k の候補集合 C_k を作成することにより候補を絞り込み効率的に同時関係抽出を行うことができる．

3.2 多次元同時関係

多次元同時関係 [8] では，距離概念を加え次元 (トランザクション) を越えたルールの抽出を行う．$n_i = \langle v \rangle$ と $n_j = \langle u \rangle$ を階 1 の空間上の 2 点 (アドレス) としたとき，n_i と n_j の相対距離は $(n_i, n_j) = \langle u - v \rangle$ となる．基準点 0 における相対距離 (相対アドレス) は $(0, n_j) = (n_j) = \langle v \rangle$ と記述される．階 1 空間上の点 Δ_j におけるアイテム i_k は拡張アイテムと呼び $\Delta_j\,(i_k)$ と表す．同様に点 Δ_j におけるトランザクション T_K は拡張トランザクションセットと呼び $\Delta_j\,(T_K)$ と表す．I_e は階 1 空間内に出現し得る全ての拡張アイテムセット，D_e を出現し得る全拡張トランザクションセットとする．

Lu [8] らは，従来のデータベースから拡張トランザクションデータベースに変換することで距離概念を追加している．表 1 に拡張トランザクションの変換例を示す．

表 1　拡張トランザクションと拡張アイテム

トランザクション	アイテム	拡張トランザクション	拡張アイテム
T1	a,b,c,d	Δ 0(T1)	Δ 0(a) , Δ 0(b) , Δ 0(c), Δ 0(d)
T2	a,b,d	Δ 1(T2)	Δ 1(a), Δ 1(b), Δ 1(d)
T3	a,c,d	Δ 2(T3)	Δ 2(a), Δ 2(c), Δ 2(d)
T4	b,d	Δ 3(T4)	Δ 3(b), Δ 3(d)
T5	b,c,d	Δ 4(T5)	Δ 4(b), Δ 4(c), Δ 4(d)

この拡張トランザクションに含まれる拡張アイテムはどのトランザクションを基準とするかにより距離が変化する．そのため，拡張トランザクションには拡張アイテムリストと起こり得る全ての相対アドレスを保持しておく．

ここで，相対アドレスの範囲について述べる．相対距離の範囲を指定しない場合，データベース内の全ての距離を記憶し全ての項目に対してルールを考える必要がある．しかし，ある一定の距離を越えると距離概念の有効性は保証されなくなる．例えば，

A 社の株価が上がれば ⇒400 日後に B 社の株価が上がる

これは，偶発したものであり同時関係として適さない．ある一定の範囲内のみに限定し無駄なルール生成を防ぐために maxspan を用いる．maxspan(ms) は，ユーザ定義の閾値であり，この範囲内のアイテムと相対アドレスを観測する．よって，支持度，確信度は

$$support = \frac{\text{条件 } X, Y \text{ を同時に満たす } ms \text{ 内のトランザクション}}{ms \text{ 内のトランザクション数}} \quad (5)$$

$$confident = \frac{\text{条件 } X, Y \text{ を同時に満たす } ms \text{ 内のトランザクション}}{\text{条件 } X \text{ を満たす } ms \text{ 内のトランザクション}} \quad (6)$$

となる．多次元同時関係の摘出アルゴリズム E-Apriori [8] を Algoritm1 に示す．

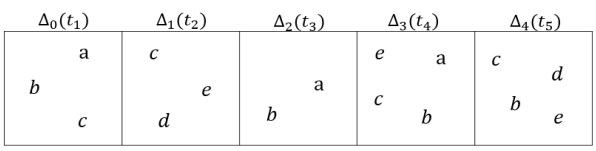

図 10　多次元同時関係

全ての距離のアイテム集合をカウントし，最小支持度，最小確信度と比較することで 1 項目頻出集合を得る．ただし，このときの支持度，確信度は厳密な値ではない．例えば，図 10 の $\Delta_0\,(a), \Delta_1\,(c) \Rightarrow \Delta_3\,(e)$ を考えるとき既存の支持度, 確信

度は $support = \frac{1}{5}$, $confident = \frac{1}{3}$ となる．しかし，厳密には $\{\Delta_0(t_1), \Delta_1(t_2), \Delta_2(t_3), \Delta_3(t_4)\}$,
$\{\Delta_1(t_2), \Delta_2(t_3), \Delta_3(t_4), \Delta_4(t_5)\}$ の 2 範囲を見ているため，$support = \frac{1}{5-3}$, $confident = \frac{1}{3-2}$ となる．つまり support,confidence はルールの間隔スパン $(\Delta_3 - \Delta_0)$ によって変化する．ところが Lu [8] らにより最大トランザクション間隔と全トランザクションの合計数を比較するとトランザクション合計数の方がはるかに大きいため上記の計算の差はごくごく小さくなり，式 5，6 を近似的に用いることができることが示されている．これにより，アイテムセットの support に重要な逆単調性も維持する．

次に 2 項目集合では，頻出項目集合から基準 $0''0'0$ のアイテムとのペアを全て候補とする．データベースを読み込み候補集合と同じアイテム・相対アドレスを持つものをカウントし，最小支持度，最小確信度を越えるものを頻出項目とする．3 項目集合以降も同様に候補を作成し，カウントをとるが，ここではハッシュを用いることで効率的に候補の絞り込みを行っている．

4. テンソルによる多次元同時関係

本章では，テンソルデータモデルの枠組みで多次元同時関係を抽出化する方法について述べる．

4.1 テンソル多次元同時関係抽出操作

多次元同時関係抽出操作は，要素のベクトル化，右シフト操作，AND 操作，カウント操作を用いることで実現する．テンソルデータ操作による多次元同時関係を抽出の様子を Algoritm2 に示す．

ここでは，3 階のテンソルについて言及する．まず，要素別ベクトル化により各要素の関係性を見れるよう変換を行う．

1 項目集合 $\Delta_{0''0'0}\{i_n\}$ では，要素別ベクトルに対して全要素 1 のベクトルとの内積カウント操作を行い，結果が最小支持度･最小確信度以上となるものを抽出する．$\Delta_{s''s's}\{i_n\}\,(s''s's \neq 0''0'0)$ では，要素別ベクトルを $x'' = s'', x' = s', x = s$ だけ右シフトを行い，得られたベクトルに対して内積によるカウント操作を行う．全ての距離について内積をとり，1 項目頻出項目集合とする．

2 項目集合以上の場合を考える．簡単のためここでは 2 項目集合について述べる．1(k-1) 項目頻出集合から候補集合 $C_2(C_k)$ を作成する．C_2 の項目集合と一致するアイテムを要素別ベクトル化から取り出す．$\Delta_{0''0'0}\{a\} \Rightarrow \Delta_{0''0'1}\{b\}$ では，a ベクトルと b ベクトルを抽出し，a ベクトルを $0''0'0$ から $0''0'1$ となるように右シフトを行う．これにより違う距離のアイテムを同じ次元としてみることができ，AND 操作を行うことで同時関係をとることが可能となる．最後に全要素 1 のベクトルと内積を行う内積カウント操作を実行し，最小支持度最小確信度と比較することで頻出項目集合抽出とする．

5. 実　　験

実験では，E-Apriori と提案手法によるアルゴリズムの計算量とデータ記述長の比較を行う．また気象データひまわり 2000 を用いて標高，地域，時間的によって変動する多次元同時関係の抽出を行う．

5.1 実 験 準 備

実験に用いる気象データひまわり 2000 では，アメダス (自動気象データ収集システム) により収集された日単位データが収録されている．本実験のテンソルデータモデルは時間，地域，標高の 3 つを距離概念として用い，3 階のテンソルを構築する．地域は 8 つのグループ (次元) に分けられており北海道地方，東北地方，関東地方，北陸甲信越地方，東海地方，近畿地方，中四国地方，九州沖縄地方となっている．標高は 0〜240m までを 30m ごとに区切った 8 つの標高区間を用いる．8 つの各標高区間を満たす観測地を各地域からランダムに抜粋し，観測地をトランザクションとしている．時間は季節的な天候の変化を考慮して 4 月 1 日から 10 日までのデータを使用する．時間の maxspan を 7 とし，1 ～ 7 日, 2 ～ 8 日, 3 ～ 9 日, 4 ～ 10 日 の 4 グループデータに対して実験行う．多次元同時関係の抽出では，それぞれのグループで得られた結果から共通項を抽出する．

トランザクション内のアイテム属性はそれぞれの観測地で観測された日降水量，日照時間，最高気温，最低気温，平均風速，最大風速を用いる．日降水量は降水がなければ 0 あれば 1 で表す．日照時間は 0〜3 時間未満，3 時間以上 6 時間未満，6 時間以上 9 時間未満，9 時間以上の当てはまる部分に 1 それ以外 0 としている．表 2 にその他の要素を定量化したものを示す．表 2 で示される 25 要素のベクトルで 1 つのトランザクションを構成する．以上により得られた標高，地域，時間，要素に基づきテンソルデータの構築を行う．

表 2　要素定量化

最高気温 ($°C$)					最低気温 ($°C$)				
～0	0～10	10～20	20～30	30～	～− 10	− 10～0	0～10	10～20	20～
平均風速 (m/s)					最大風速 (m/s)				
0～3	3～6	6～9	9～12	12～	0～3	3～6	6～9	9～12	12～

既存手法 E-Apriori では，定量化により要素属性を細分化するのではなく，1 アイテム属性を 1 要素として用いる．したがって，1 トランザクションに含まれる要素は 6 アイテムとしトランザクションデータベースを構築する．

5.2 評価と考察

記述条件を合わせるため，3 項目以上の抽出を繰り返し候補生成に訂正した E-Apriori のアルゴリズムを Algorithm3 に示す．また，気象データひまわりを用いたデータ記述長を表 3 に示す．

表 3　E-Apriori とテンソルのデータ記述長

	アイテムリスト (全要素)	相対アドレス
E-Apriori	$6 \times 8 \times 8 \times 7$	$6 \times \frac{(8\times8\times7)^2-1}{2}$
Tensor	$25 \times 8 \times 8 \times 7$	0

Algorithm2 の $foreach\,(element\ \Delta_{s''s's}(N))$ の計算量はトランザクション内のアイテムを見ているため Algorithm3 の foreach($extended\ transaction\ \Delta_{s''s's}(t)$) と等しくなる．

故に計算量はともに $O\left(n^3\right)$ である．データ記述量はアイテムリストの記述と相対アドレスの記述の総和であり，テ

ンソルデータモデルを用いた方が 593,597byte 短い記述長となる．これは一般に相対アドレスの記述長が，アイテムリストの記述長より大きくなる傾向があるからである．E-Apriori による相対アドレスは，maxspan 内の全ての相対アドレスを保持する．各軸の長さ ($maxspan$) が N'', N', N の 3 つの距離を用いたとき，E-Apriori の各要素 (M) に対し $(N'' \times N' \times N) + (N'' \times N' \times N\ -1) + \ldots + 1$ 個の相対アドレスを保持する．よって，テンソルデータモデルと E-Apriori のアイテムリストの記述長差が

$$M \times \frac{(N'' \times N' \times N)^{n-1} - 1}{2} \tag{7}$$

を越えない場合テンソルデータモデルの方が優位となる．また，階が増えるごとに相対アドレスは大幅に増加する．ただし，アイテムリストの記述長は一般に E-Apriori に比べテンソルデータの方が大きくなる．例えば，本実験の日照時間について E-Apriori では，日照時間 0~3 時間のように 1 つのアイテムのみで記述するためデータは密となる．テンソルでは，日照時間 0~3,3~6,…とベクトルで保持するため疎なデータとなり余分なデータを保持する．処理操作に違いはあるが同じ計算量で同じ同時関係抽出が行える．支持度 50%，確信度 80% で同時関係抽出を行ったとき，得られた同時関係数と例を以下に示す．

表 4　取得同時関係

	1 項目	2 項目	3 項目	4 項目	5 項目	6 項目	7 項目	8 項目	9 項目	10 項目
1~7 日	93 個	422 個	919 個	926 個	463 個	153 個	44 個	10 個	0 個	0 個
2~8 日	84 個	357 個	701 個	660 個	315 個	93 個	21 個	3 個	0 個	0 個
3~9 日	83 個	393 個	821 個	808 個	409 個	148 個	58 個	24 個	7 個	1 個
4~10 日	75 個	311 個	601 個	532 個	196 個	29 個	0 個	0 個	0 個	0 個
共通項	73 個	267 個	442 個	335 個	109 個	15 個	0 個	0 個	0 個	0 個

- 金沢の最高気温が 10 ~ 20 度
  ⇒ 次の日の金沢の最低気温は-10 度以下
- 糠内 (北海道十勝支庁) の最高気温が 0 ~ 20 度，
  六ヶ所 (青森県) の最低気温が-10 度以下
  ⇒ 次の日の糠内の最低気温は-10 度以下

得られた同時関係の内，札幌に関するルールが 671 個，函館に関するルールが 633 個，秋田に関するルールが 200 個，仙台が 904 個，六ケ所が 671 個，東京が 113 個，横浜が 240 個，金沢が 20 個となっている．同時関係が得られたことから使用したデータ期間では，北海道地方の札幌 · 函館，東北地方の秋田 · 仙台，関東地方の東京 · 横浜，北陸甲信越地方の金沢は安定した天候であることが言える．また，時間的なルールは同日，1 日後，2 日後の 3 種の組み合わせからできており天候が影響する範囲 (予測できる範囲) は 2 日間までと考えられる．

6. 結　　論

テンソルデータモデルにおける新たなデータ操作方法として要素別ベクトル化，シフト操作, 論理演算，カウント操作の提案を行った．テンソルデータモデルの枠組みの中で多次元同時関係抽出する手法を提案し，単純なデータ操作の組み合わせでこれを実現した．これにより，多次元同時関係を抽出する際問題であったデータ記述長の削減を可能とした．実データ (気象データひまわり) を用いて従来の方法と比較したとき，既存手法と比べて 593,597byte の記述が抑えられた．さらに同データを用いて支持度 50%，確信度 80% で多次元同時関係の抽出を行い，複数の意味を有するトランザクションを跨いだ同時関係 1211 件を取得した．

文　　献

[1] Agrawal, Rakesh, Tomasz Imielinski, and Arun Swami. "Database mining: A performance perspective." IEEE transactions on knowledge and data engineering 5.6 (1993): 914-925.

[2] Gao, Chuancong, and Jianyong Wang. "Direct mining of discriminative patterns for classifying uncertain data." Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2010.

[3] Han, Jiawei, Jian Pei, and Micheline Kamber. "Data mining: concepts and techniques. " Elsevier, 2011.

[4] De Lathauwer, Lieven, Bart De Moor, and Joos Vandewalle. "A multilinear singular value decomposition." SIAM journal on Matrix Analysis and Applications 21.4 (2000): 1253-1278.

[5] Kaski, Samuel. "Dimensionality reduction by random mapping: Fast similarity computation for clustering." Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on. Vol. 1. IEEE, 1998.

[6] Kolda, Tamara G., and Brett W. Bader. "Tensor decompositions and applications." SIAM review 51.3 (2009): 455-500.

[7] Lu, Hongjun, Jiawei Han, and Ling Feng. "Stock movement prediction and n-dimensional inter-transaction association rules." 1998 ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, Seattle, WA, USA, ACM, New York, USA. 1998.

[8] Lu, Hongjun, Ling Feng, and Jiawei Han. "Beyond intra-transaction association analysis: mining multidimensional intertransaction association rules." ACM Transactions on Information Systems (TOIS) 18.4 (2000): 423-454.

[9] Nguyen, Kim-Ngan T., et al. "Multidimensional association rules in boolean tensors." Proceedings of the 2011 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2011.

[10] Sun, Jimeng, Dacheng Tao, and Christos Faloutsos. "Beyond streams and graphs: dynamic tensor analysis." Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2006.

[11] Sanguansat, P., "Higher-Order Random Projection for Tensor Object Recognition.", Intn'l Symp. on Communications and Information Technologies (ISCIT). IEEE, 2010.

[12] Yokobayashi, Ryohei, and Takao Miura. "Modeling random projection for tensor objects." Proceedings of the International Conference on Web Intelligence. ACM, 2017

[13] 喜連川優. "データマイニングにおける同時関係抽出技法 (〈特集〉大規模データベースからの知識獲得)." 人工知能学会誌 12.4 (1997): 513-520.

[14] 山西健司. "記述長最小原理の進化: 基礎から最新の展開." 電子情報通信学会 基礎・境界ソサイエティ Fundamentals Review 10.3 (2017): 186-194.

$\bullet k = 1$

$C_1 = \{\{\Delta_x\,(i_n)\}\} \mid (i_n \in D) \wedge (0 \leqq x \leqq maxspan)$

foreach *extended transaction* $\Delta_s\,(t)$ **do**

  foreach *candidate* $c:\ \Delta_x\,(i_n) \in C_1$ **do**

    c.count++;

  end

end

$L_1 = \{c : \{\Delta_x\,(i_n)\}\} \mid (c \in C_1) \wedge (c.count \geqq support)$

$\bullet k = 2$

$C_2 = \{\{\Delta_0\,(i_m)\,, \Delta_x\,(i_n)\} \mid (\Delta_0\,(i_m) \in L_1) \wedge (\Delta_x\,(i_n) \in L_1) \wedge ((x \neq 0) \vee (x = 0 \wedge i_m < i_n))\}$

foreach *extended transaction* $\Delta_s\,(t)$ **do**

  foreach *candidate* $c:\ \Delta_0\,(i_m)\,, \Delta_x\,(i_n) \in C_2$ **do**

    c.count++;

  end

end

$L_2 = \{c : \{\Delta_0\,(i_m)\,, \Delta_x\,(i_n)\}\} \mid (c \in C_2) \wedge (c.count \geqq support)$

$\bullet k > 2$

for $(k = 3; L_{k-1} \neq \emptyset; k++)$ **do**

  $C_k = E - Apriori - Gen\,(L_{k-1})$

  for $(o = 1; o \neq k; o++)$ **do**

    $G_\emptyset = E - Group\,(C_k, o)$;

    foreach *extended transaction* $\Delta_s\,(t)$ **do**

      $G_{\Delta_s(t)} = E - Group\,(C_o, \Delta_s)$;

      foreach *candidate* $c:\ \{\Delta_{x_1}\,(i_1)\,, \ldots, \Delta_{x_k}\,(i_k)\} \in G_{\Delta_s(t)}$ **do**

        c.count++;

      end

    end

  end

  $L_k = \{c : \{\Delta_{x_1}\,(i_1)\,, \ldots, \Delta_{x_k}\,(i_k)\}\} \mid (c \in C_k) \wedge (c.count \geqq support)$

end

$L = \bigcup_k L_k$

Algorithm 1: E-Apriori

$\bullet k = 1$

$C_1 = \{\{\Delta_x\,(i_n)\}\} \mid (i_n \in D) \wedge (0 \leqq x \leqq maxspan)$

elemental vectorization

foreach *element* $\Delta_{s''s's}\,(N)$ **do**

  foreach *candidate* $c:\ \Delta_{x''x'x}\,(i_n) \in C_1$ **do**

    c.dotproduct++;

  end

end

$L_1 = \{c : \{\Delta_{x''x'x}\,(i_n)\}\} \mid (c \in C_1) \wedge (c.dotproduct \geqq support)$

$\bullet k \geqq 2$

$C_2 = \{\{\Delta_{0''0'0}\,(i_m), \Delta_{x''x'x}\,(i_n)\} \mid (\Delta_{0''0'0}\,(i_m) \in L_1) \wedge (\Delta_{x''x'x}\,(i_n) \in L_1) \wedge ((x''x'x \neq 0''0'0)) \vee (x''x'x = 0''0'0 \wedge (i_m < i_n))\}$

foreach $(k = 2; L_{k-1} \neq \varnothing; k++)$ **do**

  foreach *element* $\Delta_{s''s's}\,(N) \mid (0 < s'' < x''_{maxspan}, 0 < s' < x'_{maxspan}, 0 < s < x_{maxspan})$ **do**

    for *candidate* $c:\ \Delta_0\,(i_m), \Delta_x\,(i_n) \in C_2$ **do**

      right shift$\{\{\Delta_{0''0'0}\,(i_m), \Delta_{x''x'x}\,(i_n)\} \Rightarrow \{\Delta_{max(0,x'')max(0,x')max(0,x)}\,(i_m), \Delta_{max(0,x'')max(0,x')max(0,x)}\,(i_n)\}\}$;

      and operation$\{\Delta_{max(0,x'')max(0,x')max(0,x)}\,(i_m), \Delta_{max(0,x'')max(0,x')max(0,x)}\,(i_n)\}$;

      c.dotproduct;

    end

  end

  $L_k = \{c : \{\Delta_{x_1}\,(i_1), \ldots, \Delta_{x_k}\,(i_k)\}\} \mid (c \in C_k) \wedge (c.dotproduct \geqq support)$

end

$L = \bigcup_k L_k$

Algorithm 2: テンソル多次元同時関係抽出操作

$\bullet k = 1$

$C_1 = \{\{\Delta_x\,(i_n)\}\} \mid (i_n \in D) \wedge (0 \leqq x \leqq maxspan)$

foreach *extended transaction* $\Delta_{s''s's}\,(t)$ **do**

  foreach *candidate* $c:\ \Delta_{x''x'x}\,(i_n) \in C_1$ **do**

    c.count++;

  end

end

$L_1 = \{c : \{\Delta_{x''x'x}\,(i_n)\}\} \mid (c \in C_1) \wedge (c.count \geqq support)$

$\bullet k \geqq 2$

$C_2 = \{\{\Delta_{0''0'0}\,(im), \Delta_{x''x'x}\,(in)\} \mid (\Delta_{0''0'0}\,(i_m) \in L_1) \wedge (\Delta_{x''x'x}\,(i_n) \in L_1) \wedge ((x''x'x \neq 0''0'0)) \vee (x''x'x = 0''0'0 \wedge (i_m < i_n))\}$

for $(k = 2; L_{k-1} \neq \varnothing; k++)$ **do**

  foreach *extended transaction* $\Delta_s\,(t)$ **do**

    foreach *candidate* $c:\ \Delta_0\,(i_m), \Delta_x\,(i_n) \in C_2$ **do**

      c.count++;

    end

  end

  $L_k = \{c : \{\Delta_{x_1}\,(i_1), \ldots, \Delta_{x_k}\,(i_k)\}\} \mid (c \in C_k) \wedge (c.count \geqq support)$

end

$L = \bigcup_k L_k$

Algorithm 3: 記述条件を合わせた E-Apriori